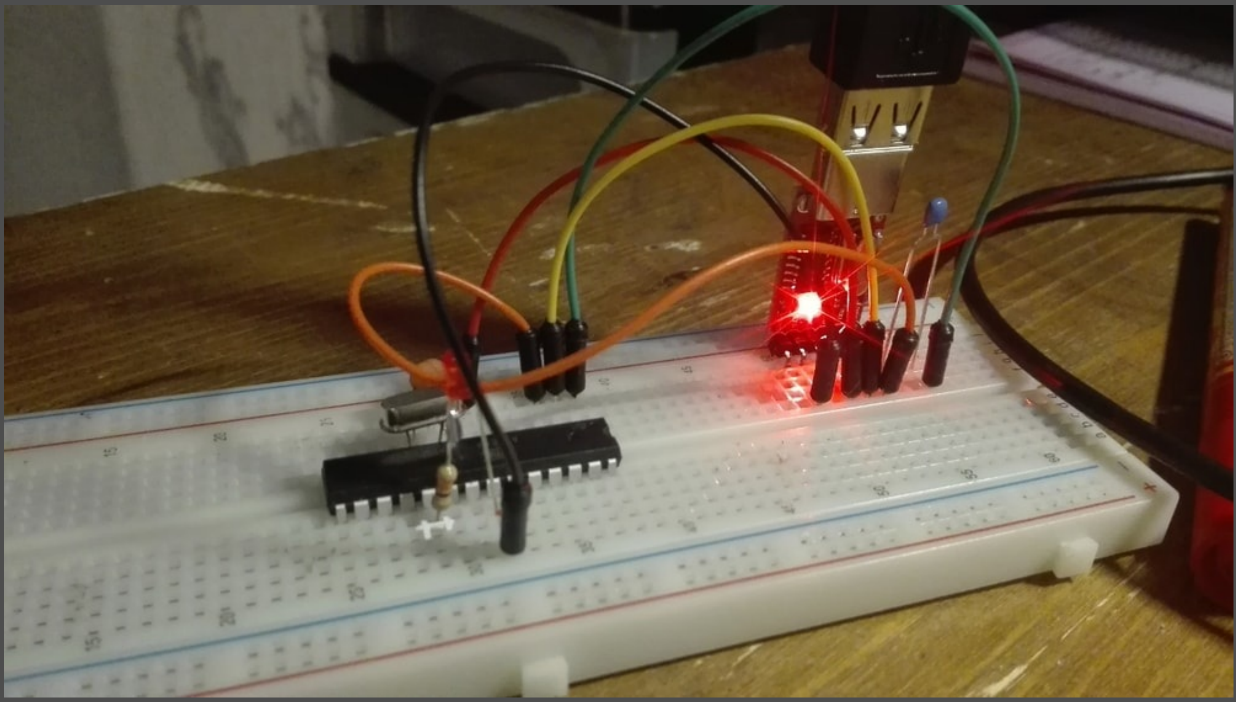


Cyberdeck V2

Preface

In most cases, a keyboard consists of several main components. These include the push buttons with the circuit board on the one hand and the microcontroller and the connection cable on the other. Since we have so far only created one board with the pushbuttons, in the next steps we have to ensure that the keyboard is also recognized as such by a computer or software and that we can send data. For this we will use an Arduino D.I.Y. microcontroller, which enables us to develop fast results in simple and small steps.

We will also not immediately create a working keyboard controller, but build an Arduino copy to adapt it to our project. In the first step we will let a light-emitting diode flash, which is like a *Hello World* of electrical engineering. This way we can be sure that everything works so far and that hardware and software will run smoothly. It is cumbersome to build a complex circuit and then find out that it does not work. If we fix errors in a basic circuit, we can concentrate better on the important sources of errors later.

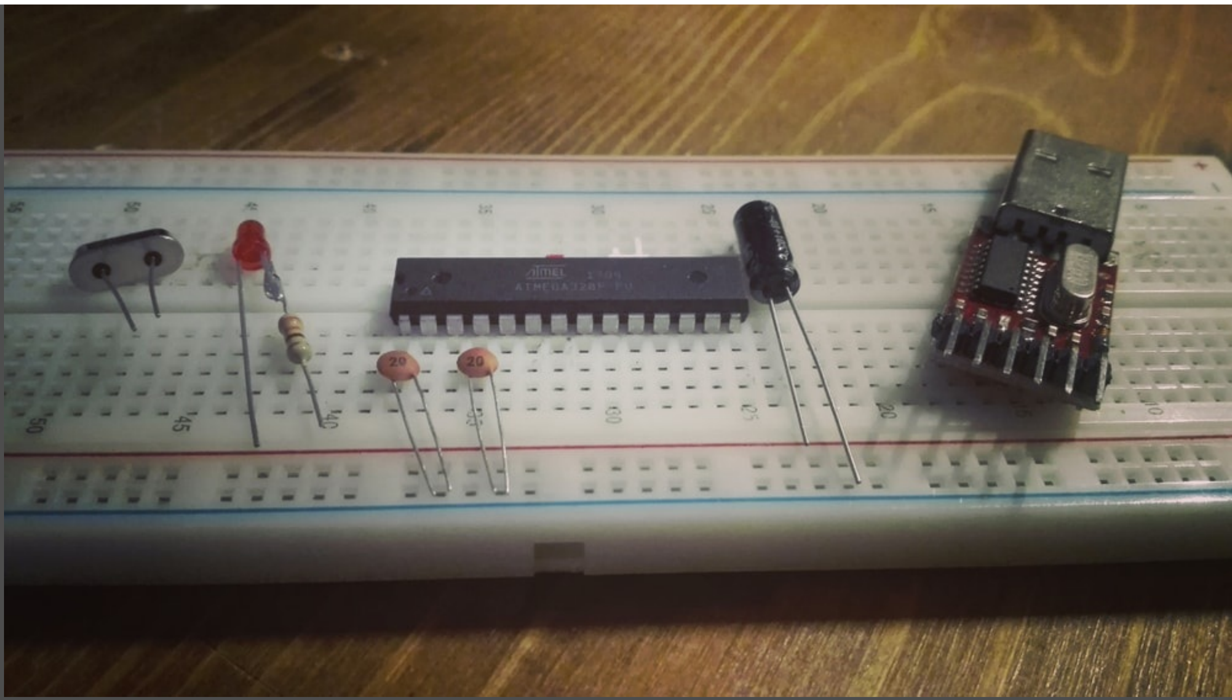
Research

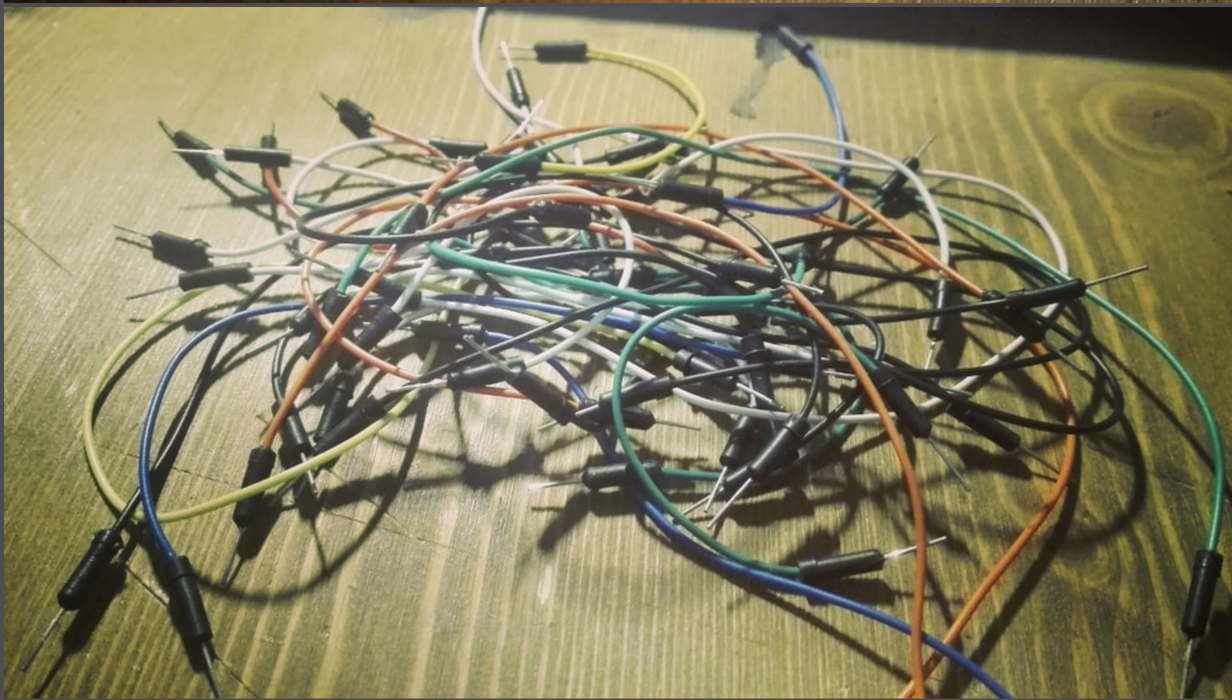
The research took place mainly on the Internet, as this is a do-it-yourself topic and is therefore more up-to-date. The main focus was on the large online video service provider, which is used by a large scene of hobby craftsmen, hackers and developers. There one could collect most of the information, compare videos with each other and choose especially according to the requirements. There were also other sources, but they were not up to date or incorrect in terms of the technical requirements.

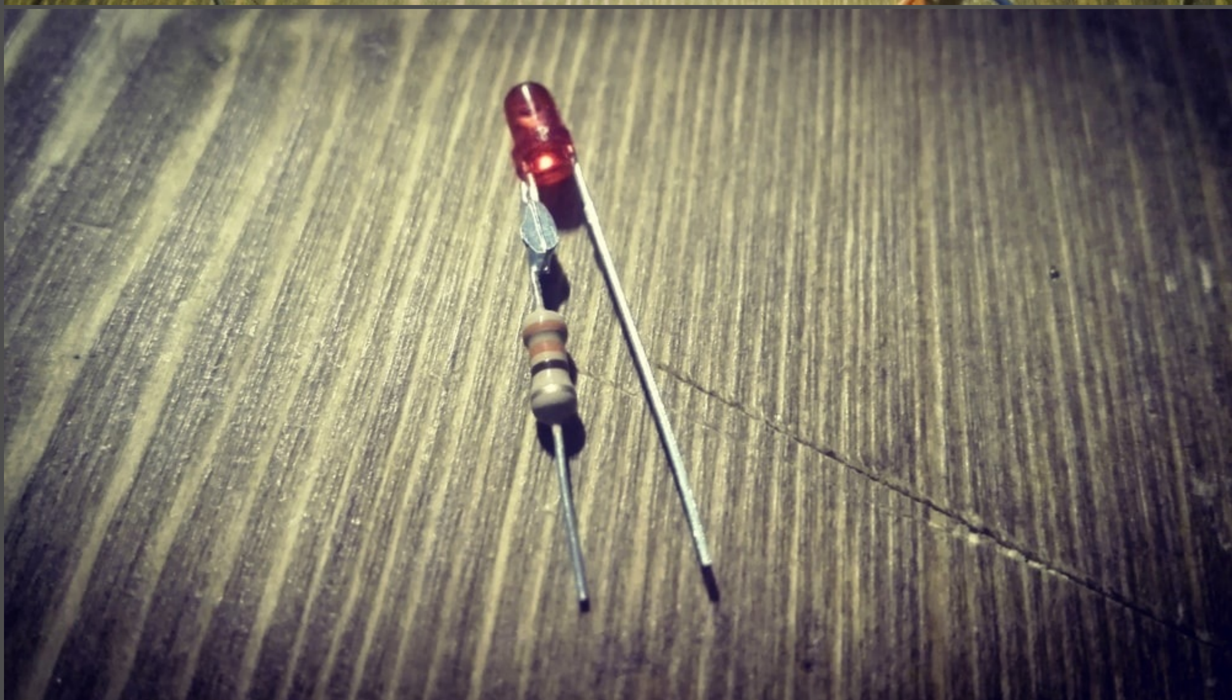
Material

- ATMEGA328P microcontroller
- USB to TTL Converter
- 100nF ceramic capacitor (104 overprint)
- 16MHz crystal oscillator
- 20pF ceramic capacitor
- LED
- Resistor
- Breadboard
- Prototype cable
- USB extension cable

All materials were ordered and delivered online by wholesalers. The delivery time of the material was not included in the time planning, since this can extend over months. You should always have a sufficient amount of material at home, otherwise projects can be very much delayed. I already mentioned this in the other documentation. I ordered the Arduino microcontroller with the USB-to-TTL converter from a major online mail order company. There are different models, so you can look around among a large number of providers.







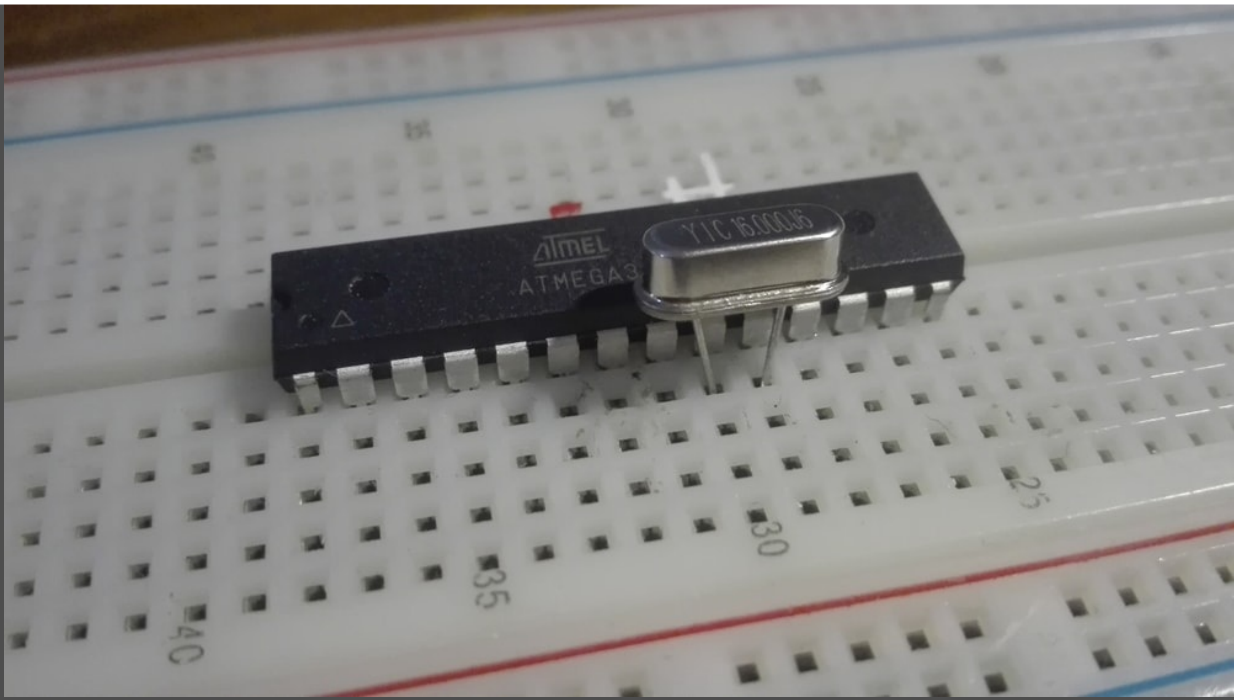
The LED with the soldered resistor saves us a lot of time when working with the pin board. Since the two components are very often used in electrical engineering, one can prepare oneself and

pre-solder some. A 220 Ohm resistor is soldered to the (anode(+)) long leg).

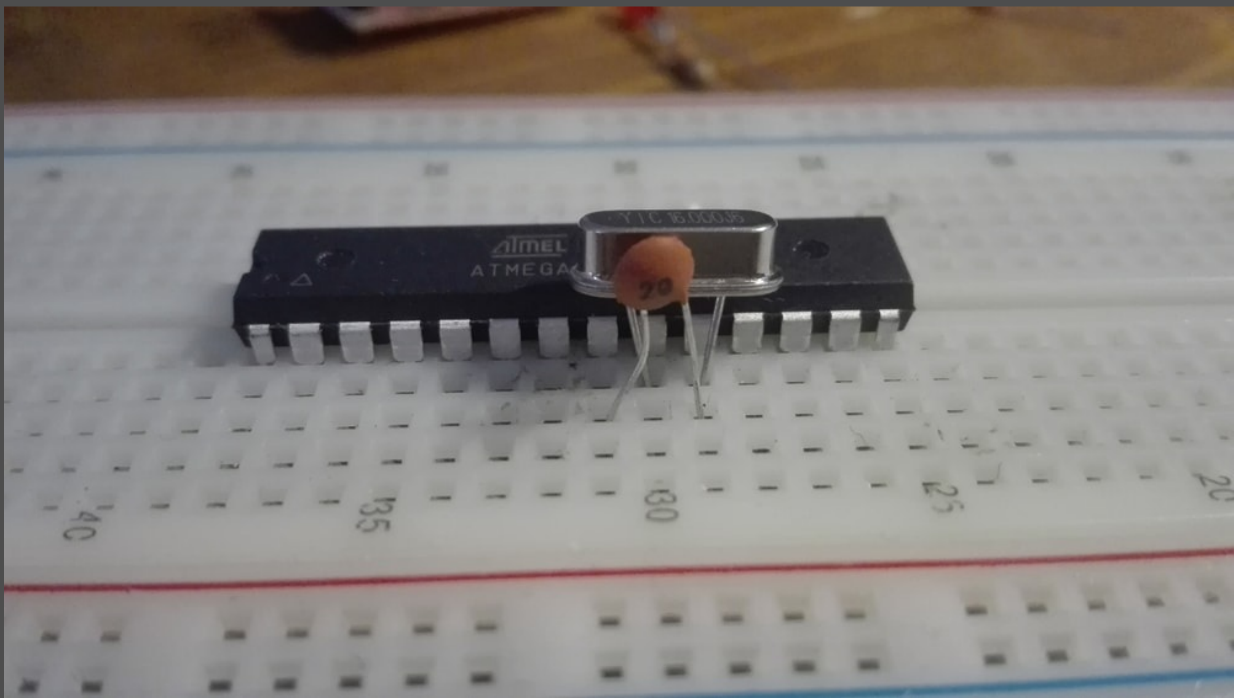
Realisation Hardware

The main component of our keyboard controller is the Atmega328P microcontroller. This has tillers, which are arranged one after the other from 1 to 28. Pin 1 starts on the side with the small triangle printed on it and the bulge, which looks like a semicircle. They are arranged counterclockwise. As you can see on the following pictures, our microcontroller is arranged so that pin 1 is on the left side in front of us. In the further work process the direction changes and this should be taken into account if you want to avoid errors in its construction.

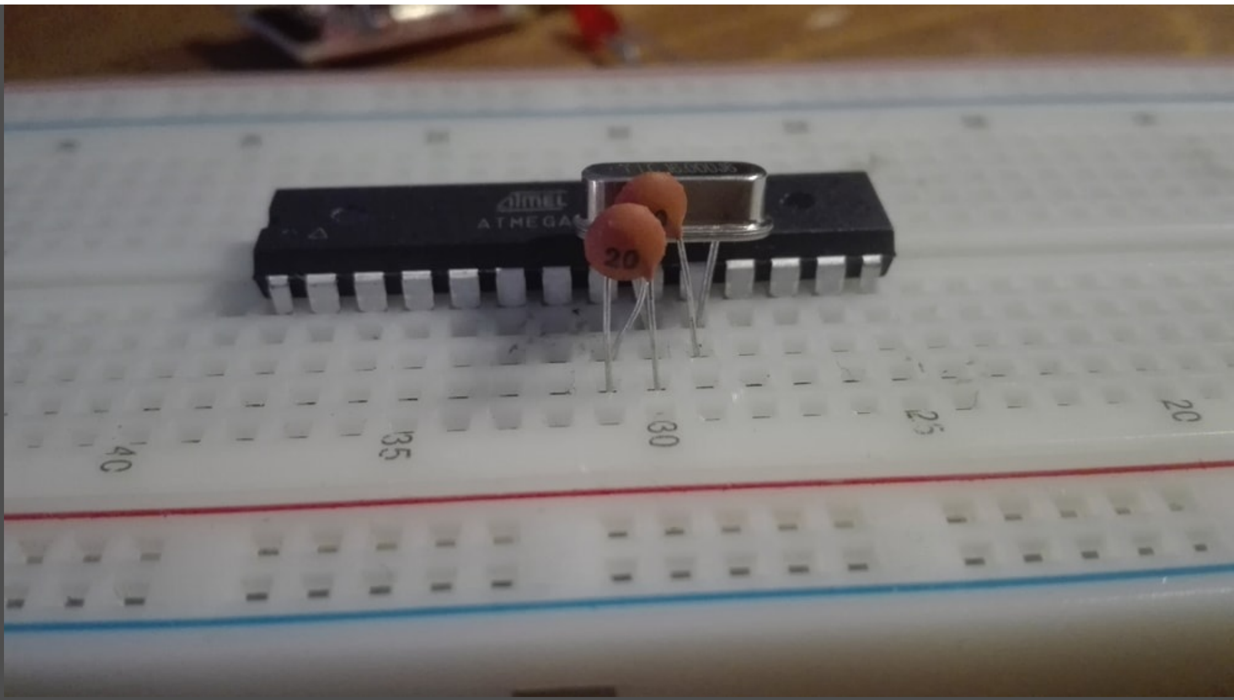
PIN Number	PIN Name	Function	Arduino Function
1	PC6	(PCINT/RESET)	Reset
2	PD0	(PCINT16/RXD)	Digital Pin 0 (RX)
3	PD1	(PCINT17/TXD)	Digital Pin 1 (TX)
4	PD2	(PCINT18/INT0)	Digital Pin 2
5	PD3	(PCINT19/OC2BINT1)	Digital Pin 3 (PWM)
6	PD4	(PCINT20/XCK/T0)	Digital Pin 4
7	VCC	-	VCC
8	GND	-	GND
9	PB6	(PCINT6/XTAL1/TOSC1)	Crystal
10	PB7	(PCINT7/XTAL2/TOSC2)	Crystal
11	PD5	(PCINT21/OC0B/T1)	Digital Pin 5 (PWM)
12	PD6	(PCINT22/OC0A/AIN0)	Digital Pin 6 (PWM)
13	PD7	(PCINT23/AIN1)	Digital Pin 7
14	PB0	(PCINT0/CLKO/ICP1)	Digital Pin 8
15	PB1	(PCINT1/OC1A)	Digital Pin 9 (PWM)
16	PB2	(PCINT2/OC1B/SS)	Digital Pin 10 (PWM)
17	PB3	(PCINT3/OC2A/MOSI)	Digital Pin 11 (PWM)
18	PB4	(PCINT4/MISO)	Digital Pin 12
19	PB5	(PCINT5/SCK)	Digital Pin 13
20	AVCC	-	VCC
21	AREF	-	Analog reference
22	GND	-	GND
23	PC0	(PCINT8/ADC0)	Analog Input 0
24	PC1	(PCINT9/ADC1)	Analog Input 1
25	PC2	(PCINT10/ADC2)	Analog Input 2
26	PC3	(PCINT10/ADC3)	Analog Input 3
27	PC4	(PCINT10/ADC4)	Analog Input 4
28	PC5	(PCINT10/ADC5)	Analog Input 5



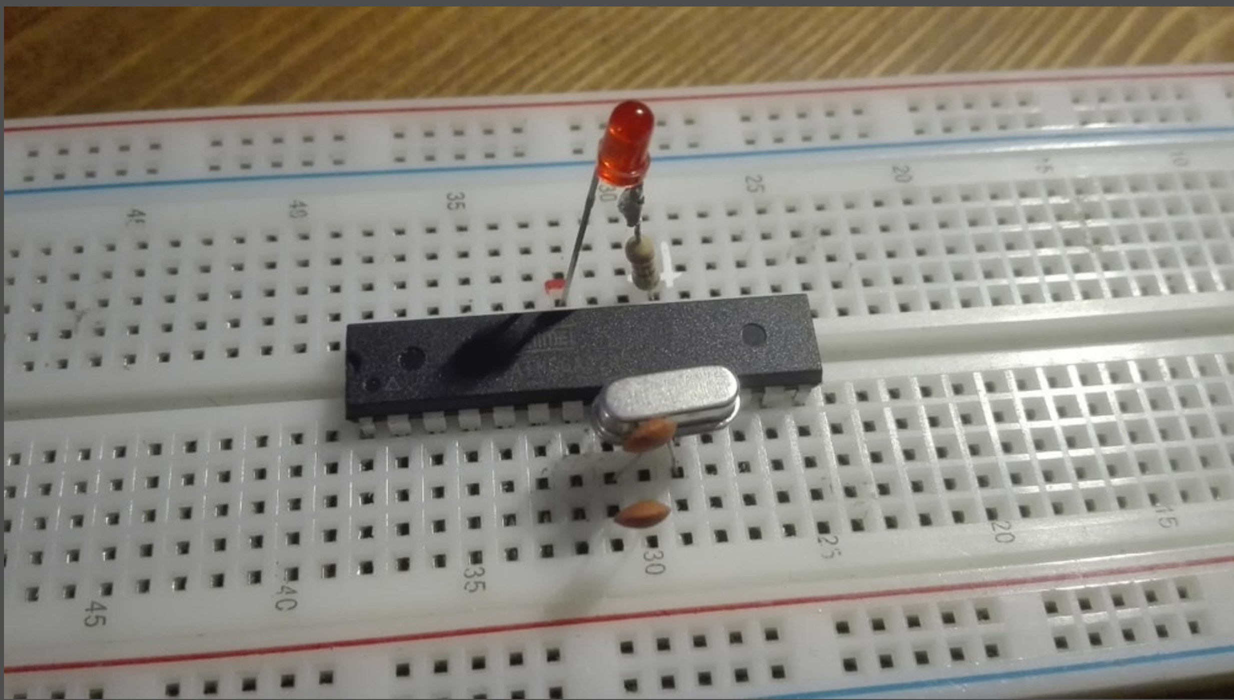
In the first step we place the quartz oscillator on tiller 9 (PB6) & 10 (PB7). It does not matter in which running direction the component is used.



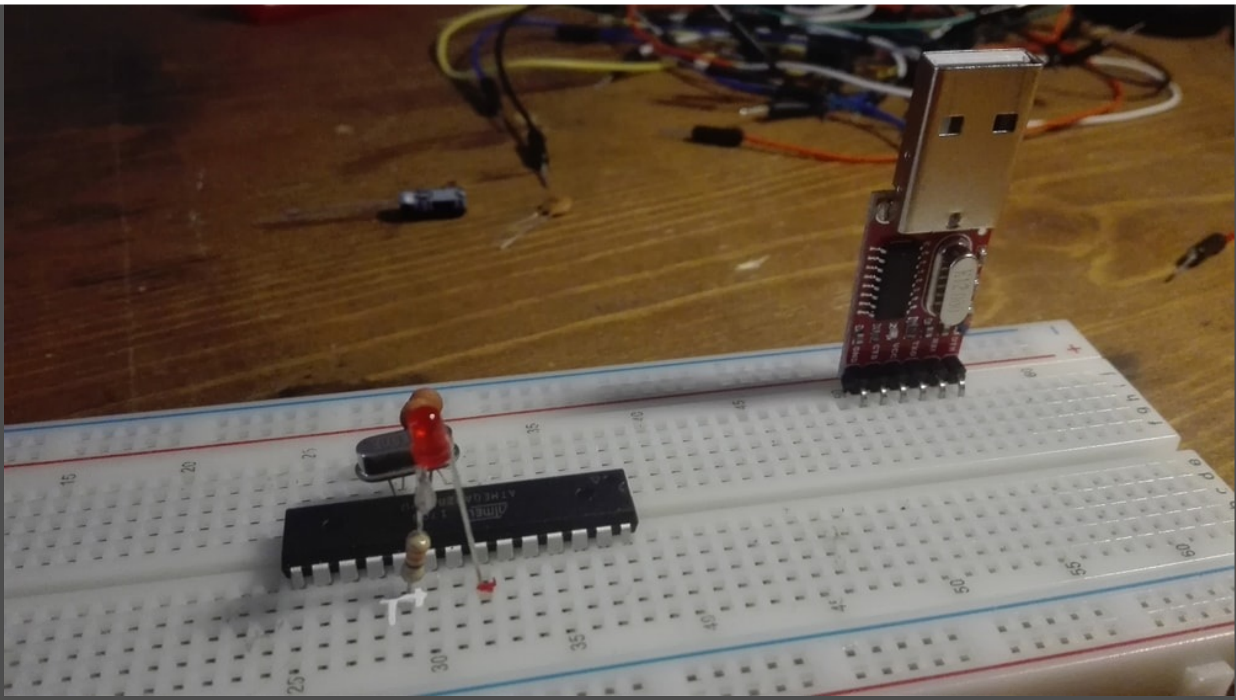
The first ceramic capacitor is connected to tiller 8 (GND) & 10 (PB7).



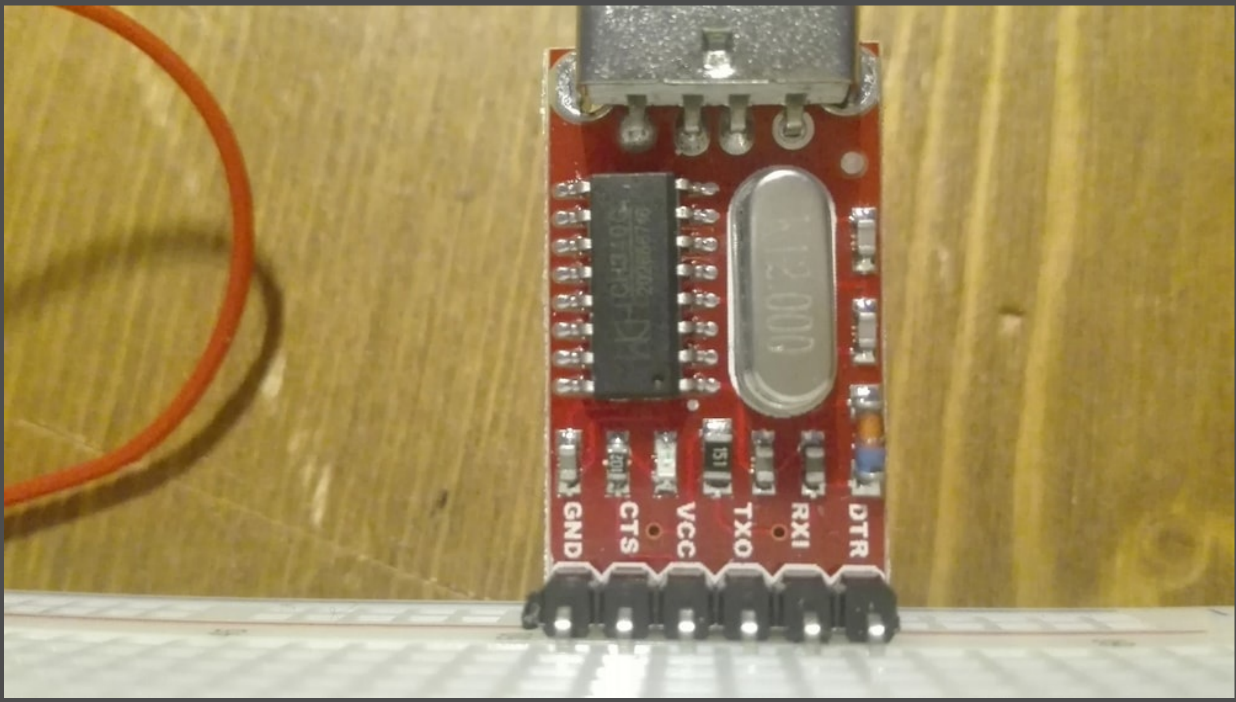
The second ceramic capacitor is connected to tiller 8 (GND) & 9 (PB6).



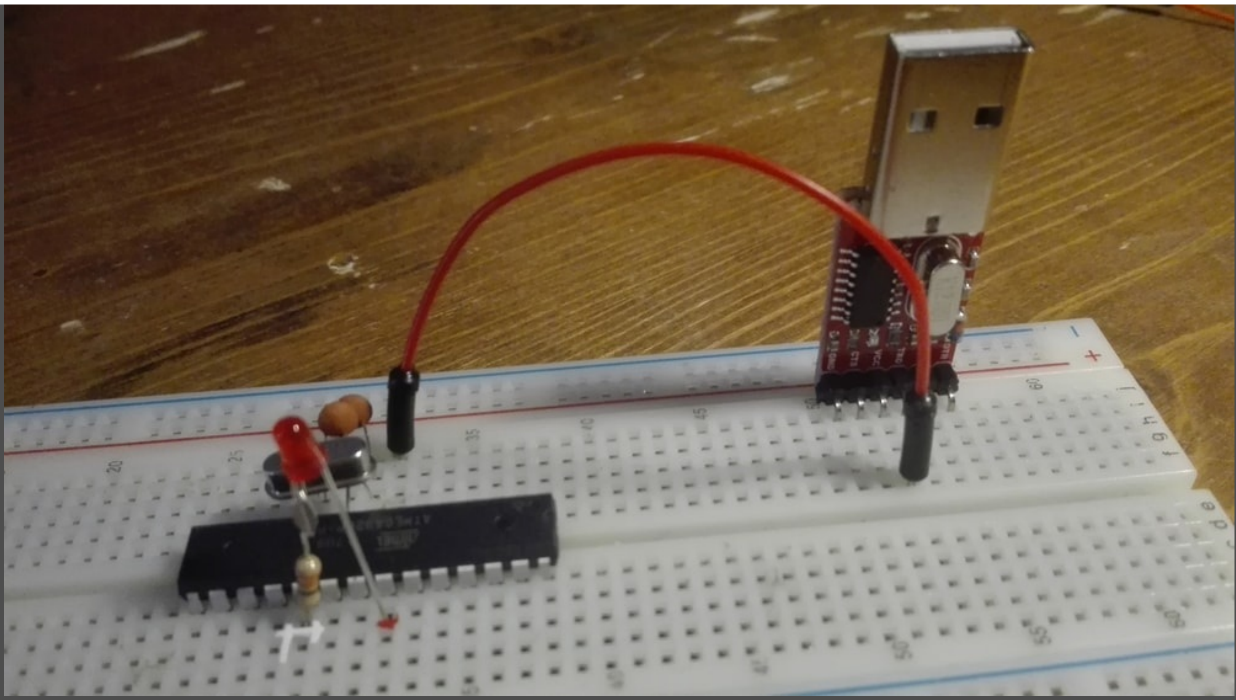
The LED is applied to pins 19 (PB15) & 22 (AVCC) on the other side, the resistor is located at pin 19 (PB15). The LED is required to perform the first function test.



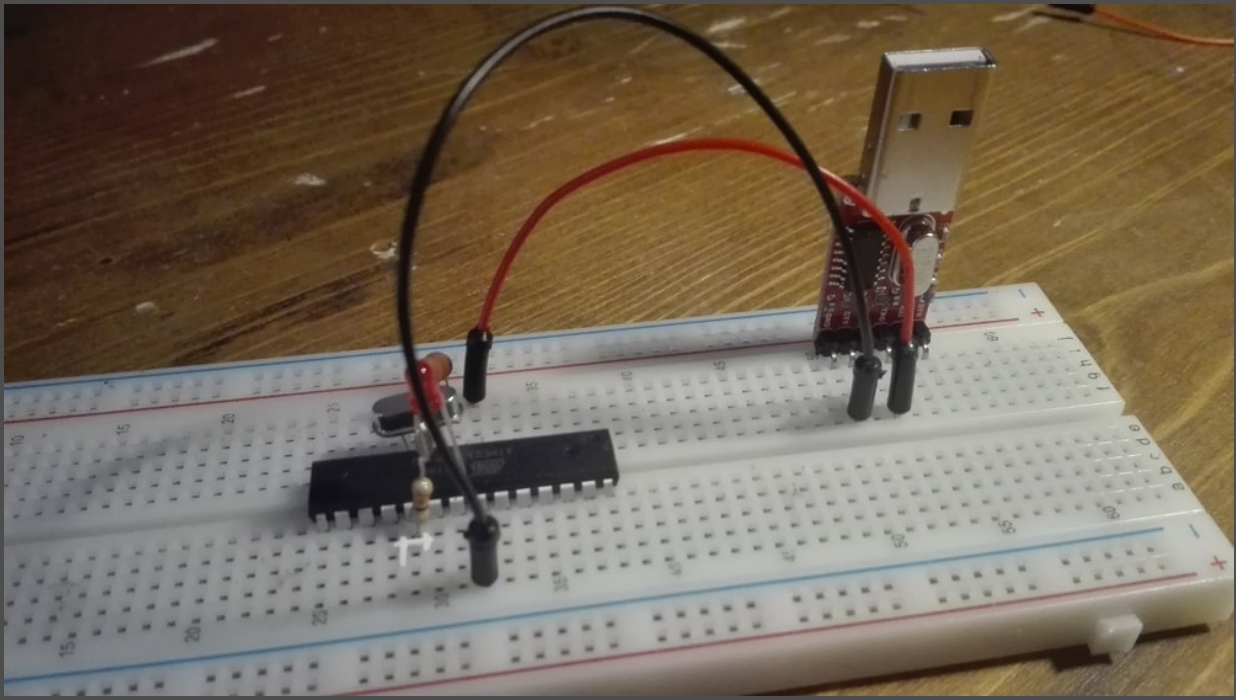
We turn the breadboard by 180° and use the USB-to-TTL converter. We take care to leave enough space for further cabling.



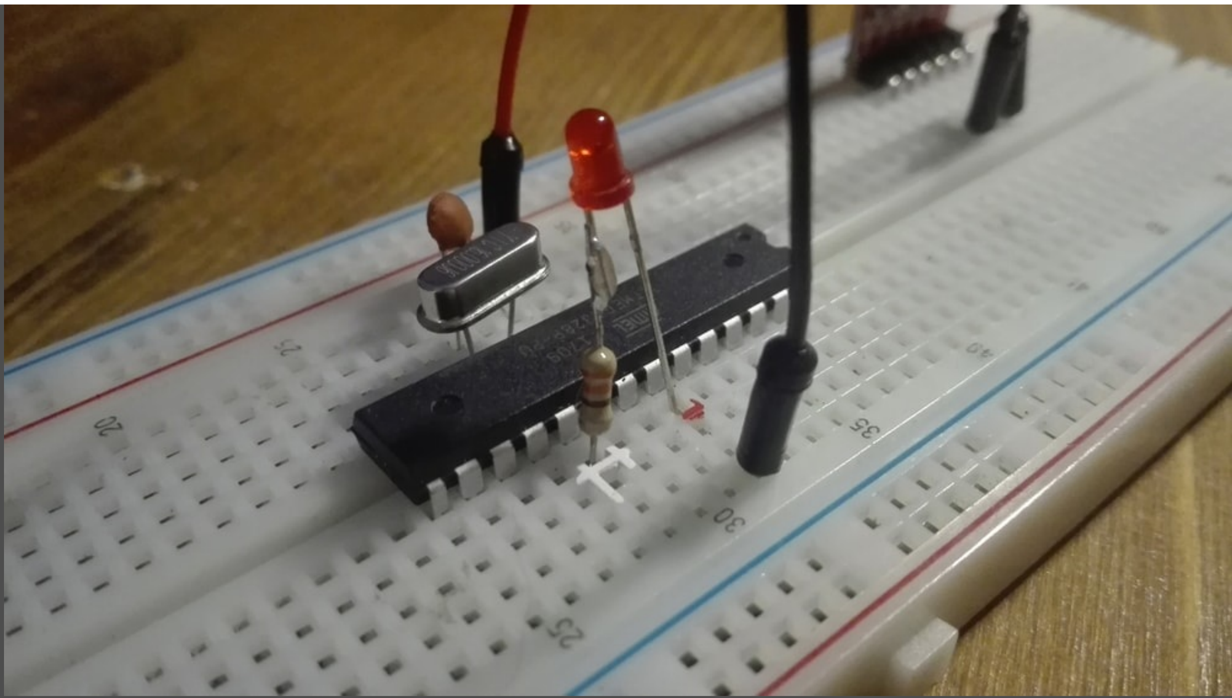
In the next steps we refer to our model, if there is another end product of converter, we have to pay attention to the labeling.



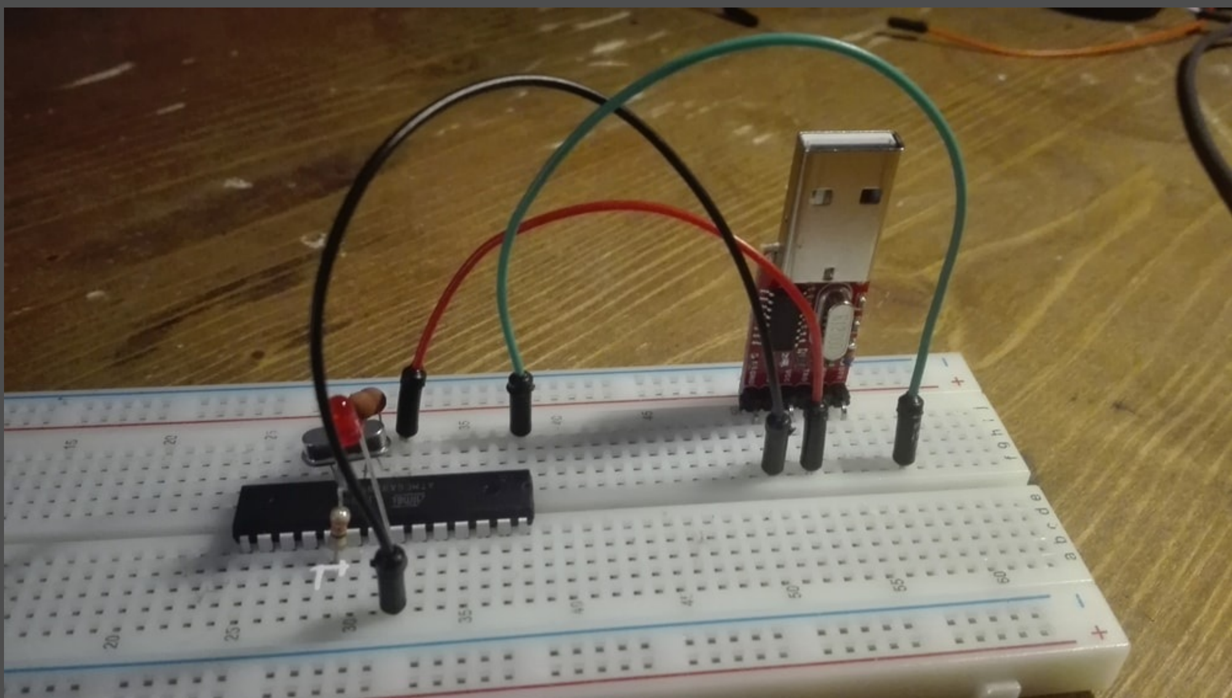
Connect the first cable (red) from VCC (positive power supply) from the converter to pin 7 (VCC) microcontroller. This ensures that our microcontroller is supplied with power.



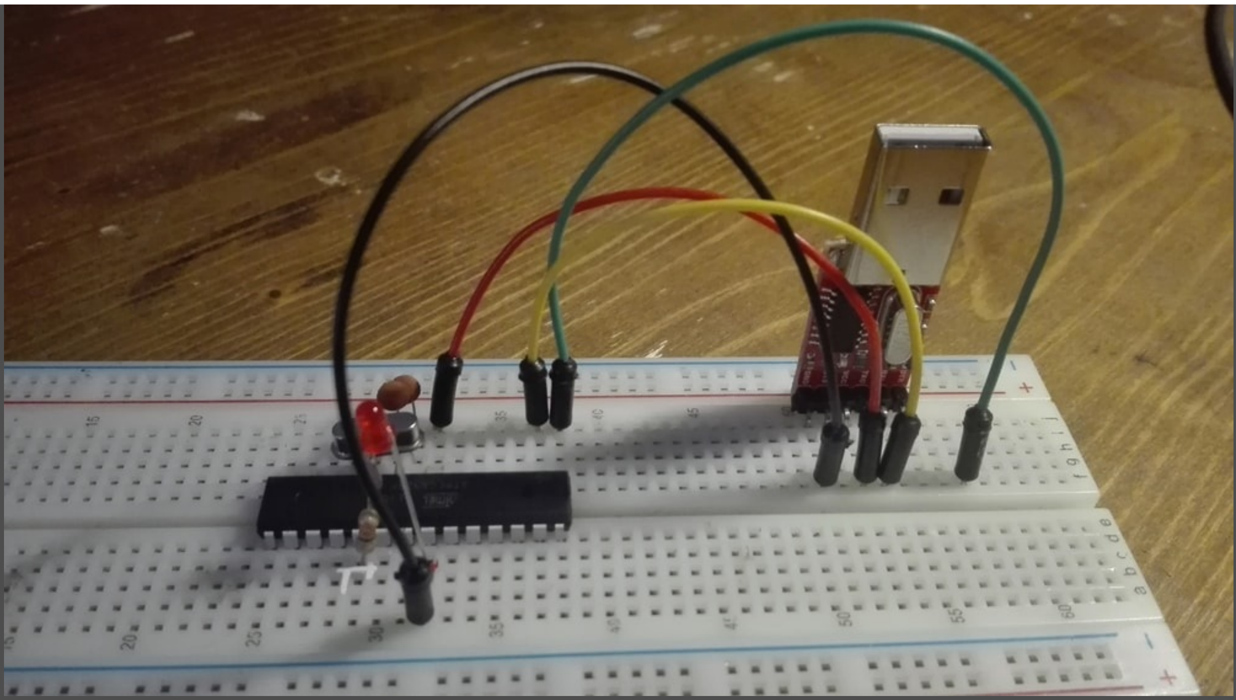
To close our power circuit, we connect the black cable to GND (converter) and pin 22 (GND).



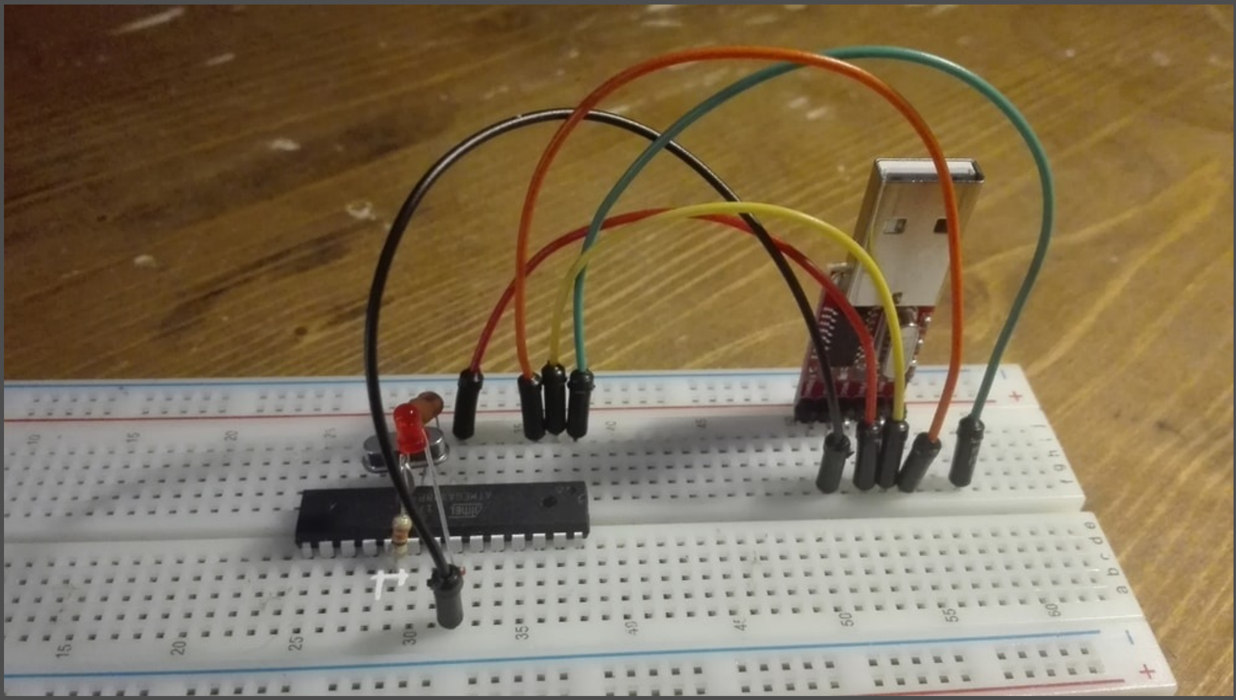
Here we see the black cable once more in detail.



Connect the green cable to pin 1 (PC6) and let the cable run to the right of our USB-to-TTL converter into the void. Why do we get explained at the end of the documentation.



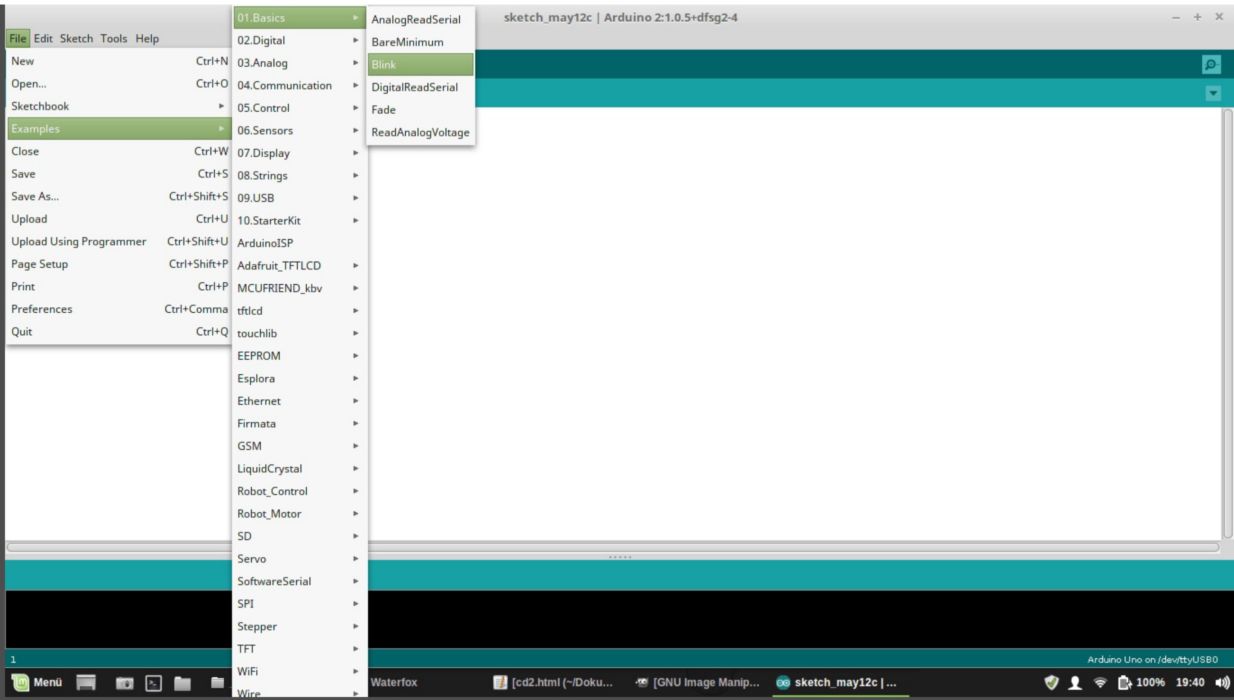
The yellow cable is connected to pin 2 (PD0) on the microcontroller and TXD on the USB to TTL converter.



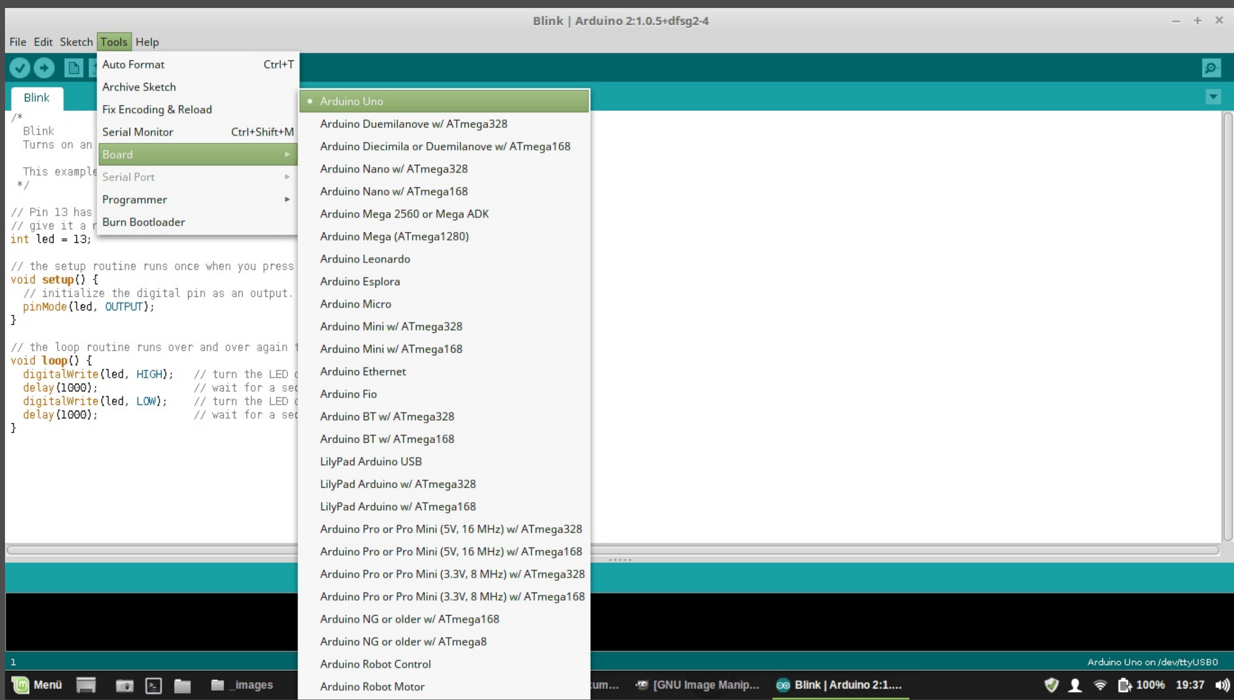
The orange cable is connected to pin 3 (PD1) and RXI. Between the green to empty running cable and the DTR pin of the converter we place another 100nf ceramic capacitor with a 104 printed on it.

Realisation Software

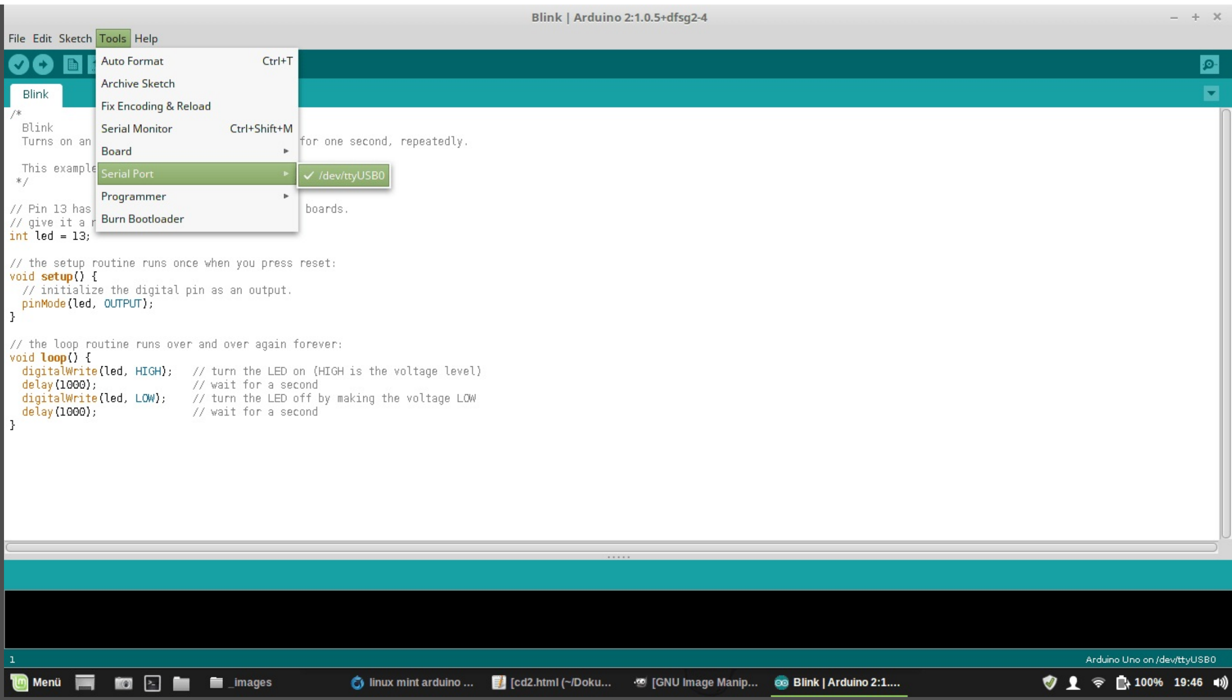
Before we can use our microcontroller to make the LED flash, we have to install software on the microcontroller. To do this, we download the Arduino IDE to our computer. You can also right-click on the pictures to enlarge them.



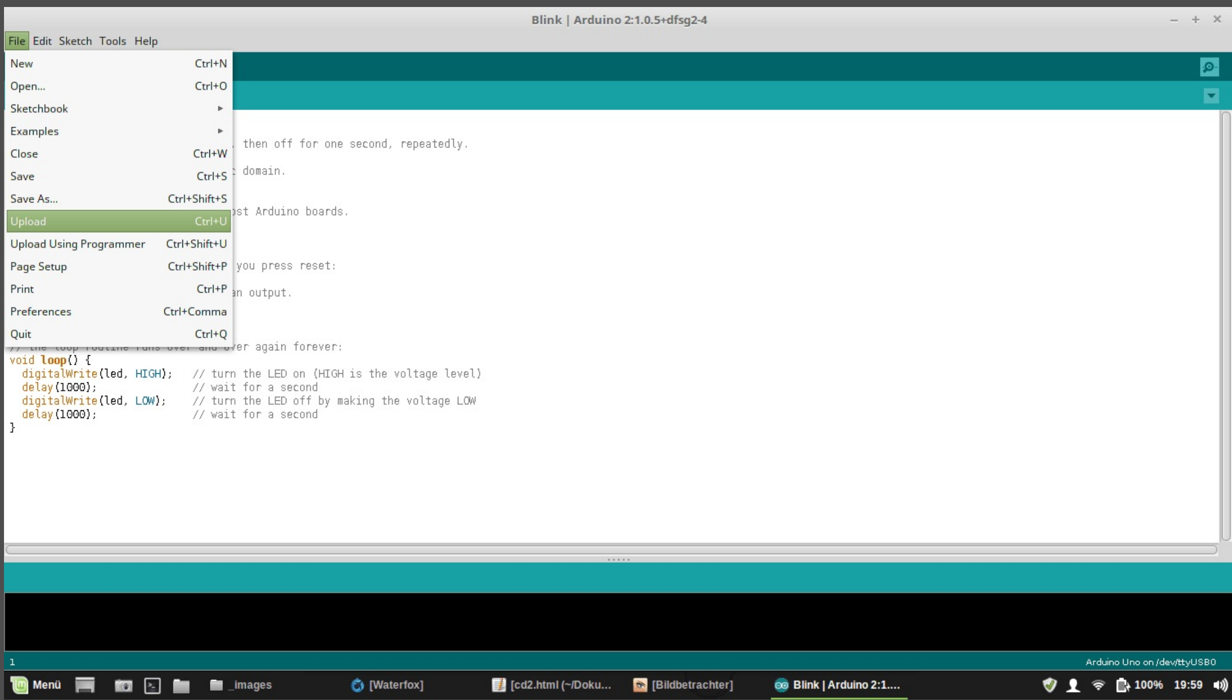
We start the Arduino software and open the flashing example. For this we follow the path `File > Examples > 01.Basics > Blink` We will not program or script ourselves, because we only want to run a first test with our built circuit. If you really want to learn about Arduino programming, you should download and read the documentation. In this project, the module is only a functional extension and in due course we will have to deal with programming again.



`Tools > Board > Arduino Uno` we select the board with which we are currently working or simulating. We have to tell this to the application, because there are a lot of different boards from Arduino and not all work the same.



`Tools > Serial Port > dev/ttyUSB0` we'll shut down our port. If you are working with Windows, there is a COM(X), where the X stands for a number. Since I work with Linux Mint, `dev/ttyUSB0` is displayed in the screenshot. Please don't let this confuse you. If the port button is grayed out, you have to connect the microcontroller with the USB cable to your computer and restart the IDE, then this should be displayed normally again.



`File > Upload (Ctrl+U)` we load the example sketch onto our microcontroller. The LED on our microcontroller starts flashing and with that we can be sure that our microcontroller or our technical structure works so far that we can remove it in the next steps and upgrade it to a working keyboard.

Conclusion

In the next steps we will attach push buttons to our circuit so that we can send certain codes to our laptop or other output device. This will initially be limited to simple functions, since we first want to prove feasibility in a study. Later our self-developed keyboard will be added to the project.